Monthly Notices
of the
ROYAL ASTRONOMICAL SOCIETY

MNRAS **485,** 5142–5152 (2019)
Advance Access publication 2019 March 6

doi:10.1093/mnras/stz638

A many-core CPU prototype of an MCAO and LTAO RTC for ELT-scale instruments

David R. Jenkins ,★ Alastair G. Basden and Richard M. Myers
CfAI, Department of Physics, Durham University, DH1 3LE, UK

Accepted 2019 March 1. Received 2019 February 28; in original form 2018 December 14

ABSTRACT

We propose a many-core CPU architecture for Extremely Large Telescope (ELT) scale adaptive optics (AO) real-time control (RTC) for the multiconjugate AO (MCAO) and laser-tomographic AO (LTAO) modes. MCAO and LTAO differ from the more conventional single-conjugate (SCAO) mode by requiring more wavefront sensor (WFS) measurements and more deformable mirrors to achieve a wider field of correction, further increasing the computational requirements of ELT-scale AO. We demonstrate results of our CPU-based AO RTC operating firstly in SCAO mode, using either Shack–Hartmann or Pyramid style WFS processing, and then in MCAO mode and in LTAO mode using the specifications of the proposed ELT instruments, MAORY and HARMONI. All results are gathered using a CPU-based camera simulator utilizing UDP packets to better demonstrate the pixel streaming and pipelining of the RTC software. We demonstrate the effects of switching parameters, streaming telemetry and implicit pseudo-open loop control (POLC) computation on the MCAO and LTAO modes. We achieve results of $<600\,\mu s$ latency with an ELT scale SCAO set-up using Shack–Hartman processing and $<800\,\mu s$ latency with SCAO Pyramid WFS processing. We show that our MCAO and LTAO many core CPU architecture can achieve full system latencies of $<1000\,\mu s$ with jitters $<40\,\mu s$ RMS. We find that a CPU-based AO RTC architecture has a good combination of performance, flexibility and maintainability for ELT-scale AO systems.

Key words: instrumentation: adaptive optics – instrumentation: miscellaneous – methods: numerical.

1 INTRODUCTION

Ground-based astronomical telescopes, both current and future, are much larger than space telescopes and so provide greater light collecting area and resolving power, increasing the potential for astronomical imaging. However due to the Earth's atmosphere, imaging from the ground without any active correction results in a significant degradation of diffraction limited imaging quality. Adaptive optics (AO; Babcock 1953) is a widely used technique that helps to negate the perturbing effects of the atmosphere and allows telescopes to achieve imaging fidelity much closer to the diffraction limit than otherwise. AO has been successfully used on sky and can be used in different configurations depending on the type and amount of correction needed.

For the next generation of extremely large telescopes (ELTs), such as the Giant Magellan Telescope (GMT; Johns et al. 2004), the Thirty Meter Telescope (TMT; Stepp & Strom 2004) and the Extremely Large Telescope (ELT; Spyromilio et al. 2008), AO correction will become much more important but also much more difficult to achieve. The atmosphere affects imaging by reducing the effective diffraction limit of ground-based telescopes by perturbing the incoming wavefront and therefore 'blurring' the image point-spread function (PSF). The relation between the strength of the atmospheric turbulence, and therefore the amount of wavefront perturbation, to the size of the PSF is known as the 'seeing' limit.

The strength of the turbulence in the atmosphere which causes the wavefront perturbations can be defined by the Fried parameter (Fried 1966), r_0, which has units of length and is usually on the order of $\sim$10 cm at a wavelength of around 500 nm. The Fried parameter is related to the size of the PSF in the seeing limit in a very similar way to the relationship of the telescope diameter to the size of the PSF in the diffraction limit. Therefore the effective diffraction limit for all telescopes with aperture diameters greater than r_0 will be constant for certain seeing conditions and roughly equal to the diffraction limit of a telescope with diameter r_0. This means that without any AO correction the resolving power of larger and larger ground-based telescopes remains effectively constant at the particular seeing limit.

★ E-mail: d.r.jenkins@durham.ac.uk

1.0.1 AO functionality

The functionality of current AO systems can be split into three main parts: indirectly detecting the incoming wavefront, reconstructing the wavefront, and then applying corrections to mitigate the effects of the atmosphere. The detection and correction of the wavefront are usually performed by optical methods, using a wavefront sensor (WFS) for the detection and a deformable mirror (DM) for correction, and the reconstruction is a computational method. The basic idea behind the reconstruction is to attempt to 'flatten' the wavefront from a point-source natural guide star (NGS) which picks up aberrations as it travels through the atmosphere. Therefore any deviations of the measured phase of the wavefront from a flat wavefront are considered perturbations and can be corrected by the DM.

For the reconstruction, analysis of the WFS images provides local wavefront gradients which are used to reconstruct the shape of the wavefront phase as projected on to the DM, this information is then used to produce commands for the DM to correct for the atmospheric perturbations. As the atmosphere is constantly changing, the corrections need to be applied very shortly after measurement for them to still be valid. This requires corrections to be performed in real time; i.e. there is a defined time limit between measurement and correction within which the reconstruction must be computed and applied such that the AO performance is sufficient for the atmospheric conditions. The real-time control (RTC) of AO is therefore fundamental to the operation of AO and is an extremely important consideration in the design of an AO instrument.

The most basic form of AO is single conjugate AO (SCAO) using a single WFS on a natural guide star (NGS) to provide wavefront measurements for a single DM. An NGS is simply a bright star that is close enough to the science object of interest such that the light from both travels through as much of the same atmosphere as possible to ensure that the reconstruction along the NGS line of sight is also valid for the science object. The corrected field of view for SCAO is small and therefore it is greatly limited by sky coverage; NGS need to be bright and close enough to the science target to reduce the effects of anisoplanatism (Fried 1982). To overcome the problem of anisoplanatism, several more complex types of AO have been proposed to allow greater sky coverage and also to increase the corrected field of view.

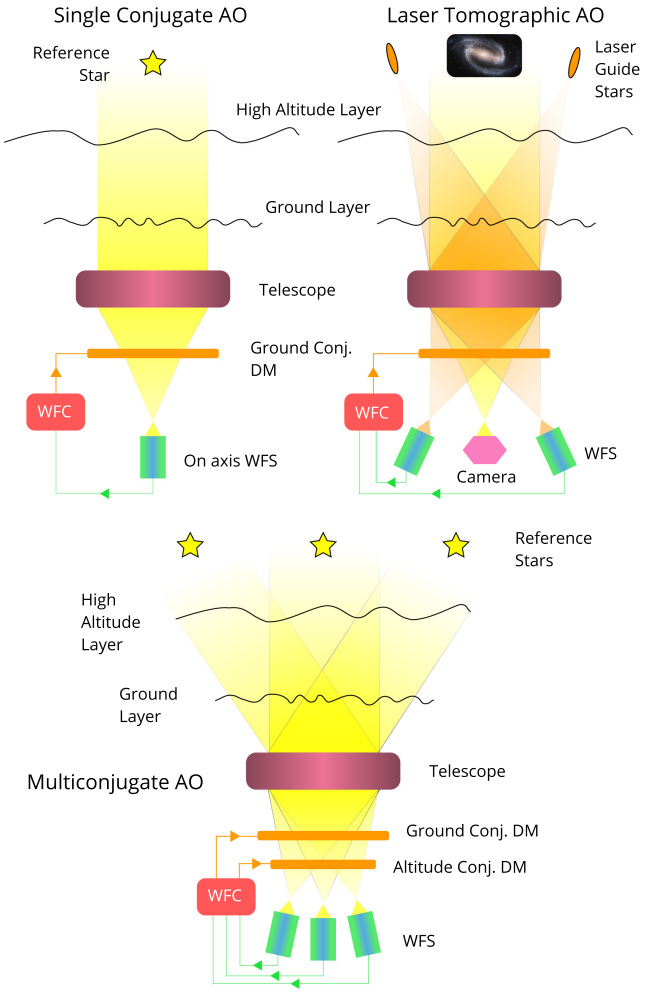

Figure 1. A visual comparison of SCAO, LTAO, and MCAO. SCAO in general has one WFS and one DM. MCAO has multiple WFSs focused on different guide stars and multiple DMs conjugated to different atmospheric layers. LTAO is similar to MCAO except it mainly uses LGS, it generally only has one DM and it corrects over a narrower FoV.

1.1 Multiconjugate and laser-tomographic AO for ELTs

Multiconjugate AO (MCAO; Beckers 1988) and laser tomographic (LTAO; Foy & Labeyrie 1985; Fugate et al. 1991) differ from the single conjugate AO (SCAO), described above, by having multiple DMs and/or multiple WFSs which increase the AO functionality over SCAO by having a wider field of view, increasing PSF field stability and the ability to do science on multiple objects. Fig. 1 summarizes the differences between SCAO, MCAO, and LTAO, however the four main processes of the RTC are common to all the types of AO as shown in Fig. 2.

MCAO gets its name from having multiple DMs conjugate to different heights in the atmosphere with multiple WFSs looking at different guide stars which can be either natural guide stars (NGS) or artificial laser guide stars (LGS). LGS use lasers to project a bright source of light high up in the atmosphere. This gives greater sky coverage as there is no fundamental restriction on the laser launch direction however it introduces a new source of error known as the cone effect or focus anisoplanatism (Beckers 1988). LTAO relies mostly on LGS WFS and usually has a single DM conjugate to the ground layer of turbulence.

There is an another type of wide-field AO known as multi-object AO (MOAO; Assémat, Gendron & Hammer 2007) which also uses multiple WFSs and DMs, MOAO aims to correct the atmospheric perturbations along a line of sight for each science target. Therefore, it uses one DM per science target in an open loop configuration with multiple guide stars across the total field. Due to the increased complexity of MOAO and its open loop nature, the architecture for MCAO/LTAO proposed here will not be directly applicable to MOAO without modification and so it will not be considered here.

The computational complexity of the reconstruction depends on both the spatial frequency of the wavefront-gradient sampling of the WFS and number of degrees of freedom of the DM. These values both typically scale to the square of telescope diameter and therefore the computational complexity of the reconstruction scales to the fourth power of telescope diameter; becoming much more of a challenge for ELT-scale telescopes and especially for the more complex types of AO which can utilize multiple WFSs and DMs and also employ more demanding reconstruction algorithms. Each extra WFS and DM in MCAO and LTAO increases the number of parameters and degrees of freedom (DoF) such that the computational complex-

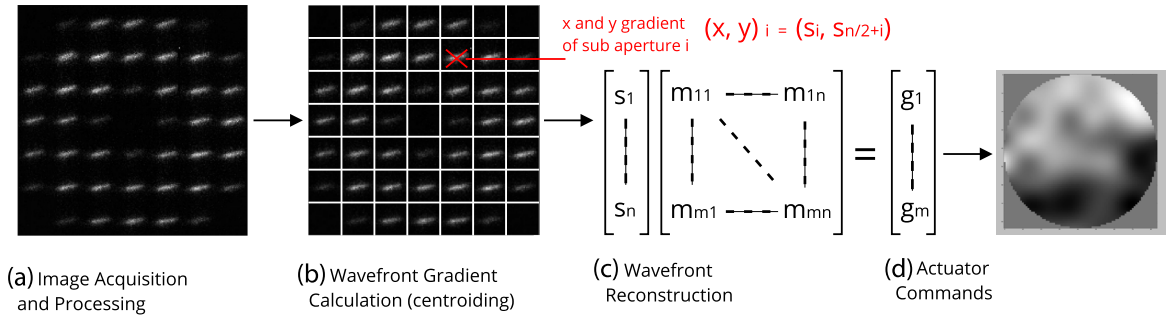

Figure 2. The four main processes in the AO loop for a single WFS. For the MCAO or LTAO case the final actuator commands from each WFS then need to be combined in a final step.

ity scales linearly with the number of each compared with an SCAO system.

Previous investigations into using many-core CPU systems, specifically the Xeon Phi, for accelerating SCAO-like AO systems can be found in Barr et al. (2015) and Jenkins, Basden & Myers (2018). This report will continue the work in Jenkins et al. (2018) and apply the method to develop a prototype system suitable for the more complex and computationally demanding MCAO and LTAO. Section 2 will give a brief update on the general optimizations and SCAO results presented Jenkins et al. (2018). Section 3 will describe the extension of this work to prototype an MCAO/LTAO system. Section 4 will present the results of this work and Section 5 will conclude with a discussion of the results and future work.

2 RECENT DEVELOPMENTS OF DARC FOR MANY-CORE CPUS

The general optimizations and modifications made to the RTC software used in this report, the Durham Adaptive Optics Real-time Controller (DARC), is detailed in Jenkins et al. (2018). However, since then improvements have been made to the software resulting in an improvement in performance and functionality. One of the main changes is a move away from the Aravis GigE Vision Library (AravisProject 2018) for simulating the camera and for receiving the pixel stream. We have instead implemented a camera simulator based on the proposed standard for ESO ELT WFS (Downing et al. 2018) that streams pixels using UDP packets, and we have also developed a library within the RTC to receive this new camera stream.

2.1 UDP camera simulator

We have developed the new ESO camera interface from the ground up, rather than relying on a pre-existing library which is not optimised for the extreme performance we require. The UDP camera simulator is controlled at runtime of the simulator executable and the receiver software simply waits to receive the packets. In this way, we can define the parameters of the camera simulator separately from the receiving of the camera stream, and it no longer requires a heartbeat thread to keep the camera sending packets, which we found could interfere with the pixel stream.

The camera simulator is implemented in the C programming language in an effort to make it both as low level as possible and as easy to modify and develop as possible. The underlying networking uses packet sockets, which are used to receive or send raw packets at the device driver (OSI Layer 2) level (Kerrisk 2018). This in theory allows minimal overhead from the kernel when sending and receiving packets as most of the protocol implementation can be programmed in user space on top of the physical layer. The type of socket used here is SOCK_DGRAM which does not have the link level header removed by the network stack and so is not quite as low level as SOCK_RAW packets.

The operating system on the simulator machine is Ubuntu 16.04 on top of a Linux 4.4.0 generic kernel. A low-latency Linux kernel was investigated but was observed to provide no discernible performance benefit. Some OS, Kernel, and network level tuning was performed to improve performance. Some of the steps taken involved:

(i) isolating most CPU cores from the OS scheduler,
(ii) specifying the core affinity for the NIC interrupts and camera threads,
(iii) tuning both the NICs and linux network stacks UDP buffer sizes using the linux 'ethtool' command and the 'sysctl' utility, and
(iv) setting the CPU power settings to 'performance'.

The camera simulator software is used only to simulate the pipelined transfer of pixels to the RTC and so the unique images that were streamed by each camera were created ahead of time via AO simulations to properly construct images for the type and dimensions of AO system tested. The images were stored on a PCIe fast raid storage array consisting of four Samsung 960 EVO NVMe solid state drives (SSDs) which provided transfer rates >4.2 GBs^{-1}, which is sufficient to allow the pixel data to be streamed directly from storage. The simulator software is set-up such that both the inter-packet delay and inter-frame delay can be set independently. This not only allows different camera frame rates to be simulated, it also allows the readout time to be adjusted to better reflect that of a real camera, rather than just sending out the packets as soon as possible. As the inter-packet delay needs to have microsecond precision the timing is achieved via the Linux 'timer_fd' utility which uses file descriptors to achieve a repeatable high-precision timer.

The simulator hardware consists of a 2012 Intel Xeon E5-2650 dual socket system with eight CPU cores and 32GB of DDR3-1600Mhz RAM per socket with a base CPU frequency of 2.0 GHz. The network devices used are two PCIe Intel Ethernet X710-DA4 Network Interface Controllers (NICs) with four 10GbE ports each where a single camera simulator stream will have exclusive use of one of these eight interfaces. In a multisocket CPU system, certain PCI-e lanes are physically connected to a single CPU socket. The NICs were therefore installed in the host such that each was local

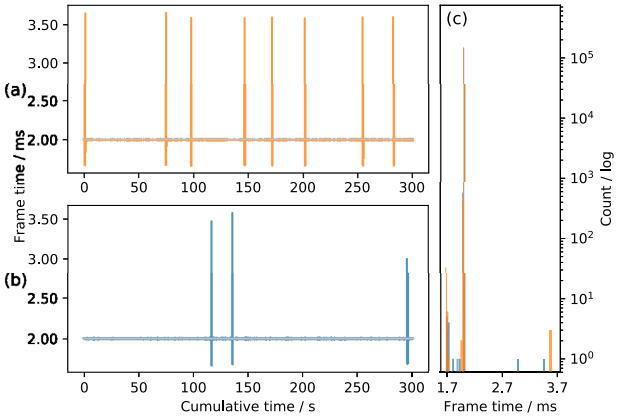

Figure 3. Frame time results from the camera simulator software showing two representative samples of frame time distributions while the camera simulator is delivering the seven individual camera streams as needed by the MCAO and LTAO RTC architectures described in Section 1.1. Results shown are for 1.5×10^5 iterations at 500 Hz for a total time of 300 s.

to a different CPU socket and then the camera threads for each interface were assigned CPU cores on the local socket.

The camera simulator is unable to provide completely jitter free cameras streams, Fig. 3 shows two representative frame time distributions from the camera simulator whilst it was delivering seven individual camera streams for an MCAO or LTAO set-up as described in Section 1.1. For 10 random distributions similar to those shown, the number of outliers can vary from 0 to 11 over the 300 s of running time, the largest outlier is never more than twice the frame time. For the periods of low jitter, the RMS jitter is very low and on order of 5 µs. Each of these data sets were collected after restarting the camera simulator software and the amount of jitter present in each run can vary significantly. This introduces random high latency spikes into some of the AO RTC timing data presented in this report. This was minimized by waiting a small amount of time to determine if a certain run was more stable than normal. Once the software was running the amount of jitter varied little and so once a stable run was found, it was used for as many RTC tests as possible.

Due to the simulator software needing to time the inter-packet delay to microsecond precision and to deliver up to seven individual camera streams at high frame-rate, the CPU system used is not an ideal candidate. The workload of the simulator is very different to that of the AO RTC system and so a more modern single socket CPU system with faster cores and lower latency memory could potentially improve the camera simulator's jitter performance. We note that the jitter originating from the camera simulator would not be present in real cameras as they are usually deterministic.

2.2 Pyramid WFS, pixel handling, and slope computation

We have also implemented a SCAO RTC demonstrator for a system using a Pyramid WFS (Pyr-WFS). Pyr-WFSs (Ragazzoni 1996) differ from the more conventional Shack–Hartman WFS (SH-WFS) in the fundamental approach to detecting the local tip/tilt slopes across the telescope aperture. In the context of the RTC, the main difference between the slope computation of a Pyr-WFS to a SH-WFS is the reduced pixel counts in the calculation of each slope measurement; only 4 pixels are needed for each slope measurement with a Pyr-WFS, whereas the SH-WFS generally

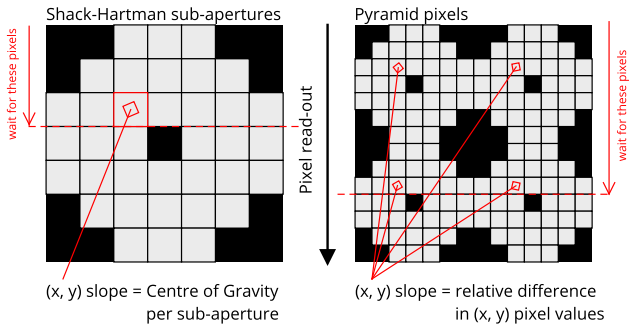

Figure 4. Comparison of the pixel pipelining for SH-WFS and Pyramid WFS image layouts. Each slope measurement from a SH-WFS is taken as the centre of gravity of the pixels values in each sub-aperture, slopes can be calculated as soon as all sub-aperture pixels have arrived. For the Pyramid WFS, the slopes are calculated as the relative x and y differences between corresponding pixels from each pupil quadrant, no slopes can be computed until at least half the frame has been read. This difference makes pipelining of slopes much less effective when using a Pyramid WFS. Here, we have assumed that the pixels are read from the top of the detector to the bottom, however the principle is the same for other read-out regimes.

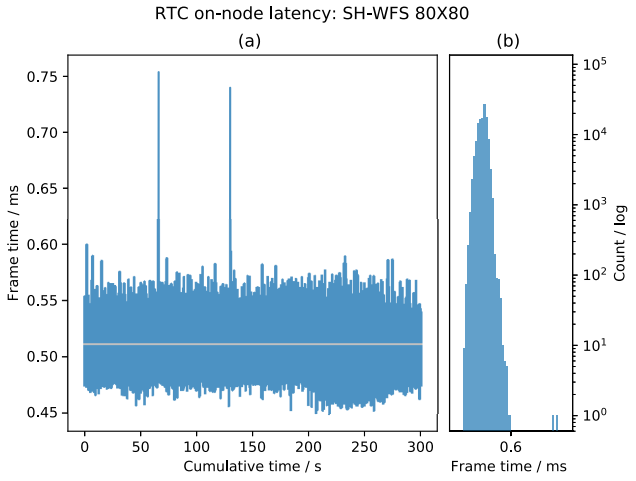

Figure 5. Frame time results for an SCAO set-up using a Shack–Hartman type WFS slope calculation with 80 sub-apertures across the pupil. Results shown are for 1.5×10^5 iterations at 500 Hz for a total time of 300 s. The mean latency is 511 ± 15 µs.

requires more (>36 for ELT-scale) depending on the specific AO systems requirements.

There is also a difference in the memory access of the pixels required for each slope measurement and in the pipelining of the pixels, shown in Fig. 4. For a Pyr-WFS the RTC needs to wait for more than half the WFS image to arrive before it can begin slope calculation whereas for a SH-WFS centroiding can begin as the soon as enough lines of pixels have arrived to complete a row of sub-apertures. This shows that even though there are far fewer pixels to process for a Pyr-WFS, the less efficient pipelining means that processing Pyr-WFSs in the RTC can be less computationally efficient than SH-WFSs.

2.3 Up-to-date SCAO results with camera simulator

Fig. 5 shows frame time and latency results for DARC running an SCAO RTC on an Intel Xeon Phi 7250 with an attached simulated

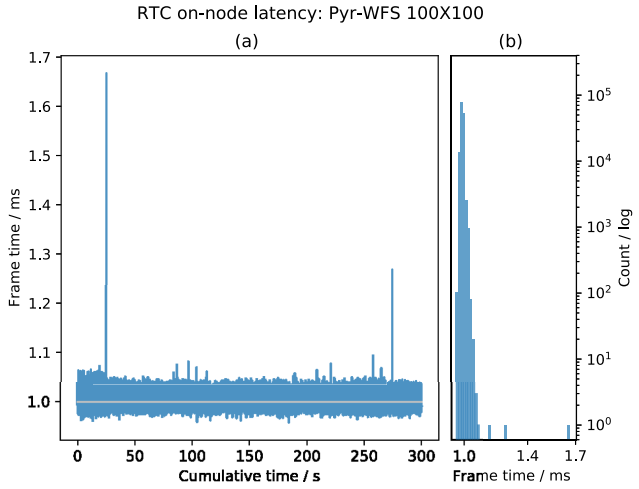

Figure 6. Frame time results for an SCAO set-up using a Pyramid type WFS slope calculation with 100 pixels across each quadrant. Results shown are for 1.5×10^5 iterations at 500 Hz for a total time of 300 s. The mean latency is 998 ± 10 µs.

camera using the UDP pixel streaming method as described in Section 2. These results are for a SCAO set-up with a single 80×80 SH-WFS with 4616 valid sub-apertures and an ELT-like M4 + M5 DM configuration with a total of 5318 actuators. The reconstruction therefore, is a single MVM of dimensions 5318×9232. There is 300 s worth of data corresponding to 1.5×10^5 frames at a frame rate of 500 Hz, the average latency is measured at 511 ± 15 µs. This compares favourably with similar results of 640 ± 20 µs from Jenkins et al. (2018), with a reconstructor of dimensions 5170×9416. The problem size used here has been adjusted to better reflect the potential dimensions of actual ELT instruments using more up-to-date information (Biasi et al. 2016; Correia 2018).

Fig. 6 shows frame time results for a similar AO set-up as above but using a Pyr-WFS instead of a SH-WFS, with the differences between the WFS processing as described in Section 2. There are 300 s worth of data corresponding to 1.5×10^5 frames at a frame rate of 500 Hz, and the average latency here is measured at 998 ± 10 µs. The dimensions of 100×100 pixels for the Pyr-WFS and 80×80 sub-apertures for the SH-WFS were chosen as these are the likely dimensions that would be used for real WFS on ELT-scale instruments. The SH-WFS dimensions result from technological limitations of the sensors being developed by ESO and the required dimensions of each sub-aperture (Schreiber et al. 2018). The Pyr-WFS dimensions were chosen based on what is being targeted for the ELT first light instrument HARMONI (Thatte et al. 2014) SCAO mode (Schwartz et al. 2018).

Figs 8 and 7 show the latency scaling of both the SH-WFS RTC and Pyr-WFS RTC against sub-apertures/pixels across the pupil. Due to the use of a simulated camera as described in Section 2, there are some unavoidable latency spikes that result from delays of the pixel transmission from the simulated camera. The mean latencies, RMS jitters and largest outliers are shown in Table 2 for data sets containing 1.5×10^5 samples at 500 Hz. Table 2 also shows the RMS jitter and largest outliers for each case for a reduced subset of the data containing at least 2×10^4 continuous iterations. These reduced sets were chosen to eliminate any major outliers resulting from camera delays to give a better idea of the 'steady' latency distribution.

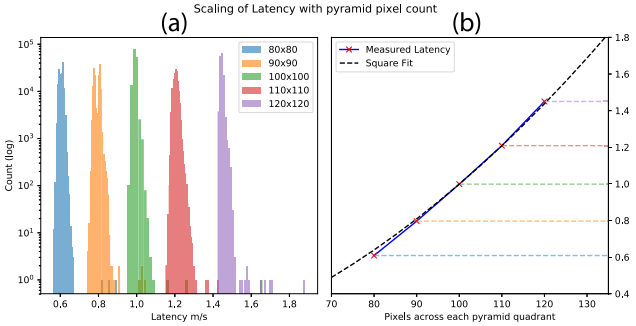

Figure 7. The scaling of latency with pixels across the pupil for the Pyramid type WFS. The latency is that for the entire RTC operation from pixels to DM commands as shown in Fig. 2, measured from the time the last pixel arrives until the DM command is ready. The relevant values for the data are shown in Table 2.

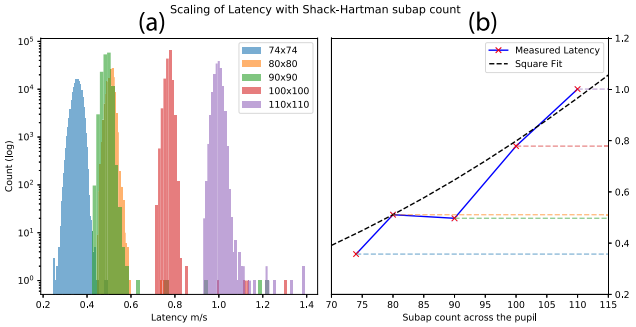

Figure 8. The scaling of latency with sub-apertures across the pupil for the Shack–Hartman type WFS. The latency is that for the entire RTC operation from pixels to DM commands as shown in Fig. 2, measured from the time the last pixel arrives until the DM command is ready. The relevant values for the data are shown in Table 2.

For all of these results, the input image sizes are kept constant for each type of WFS processing used, images of 800×800 pixels are used for the SH-WFS and 240x240 for the Pyr-WFS and either the number of pixels per sub-aperture is reduced to provide more sub-apertures or a smaller area within the input image is used for reduced numbers of sub-apertures. In this way, the pixels received are kept constant between the different WFS and only the calibration, centroiding and reconstruction are affected by the different dimensions. We can see that for the Pyr-WFS the scaling matches very closely to a square fit, which is as expected, this is because the change in degrees of freedom in the reconstruction scales to the second power of the number of pixels across the pupil.

For the SH-WFS results shown in Fig. 8, there is no clear fitting to a square fit and the case of 90×90 sub-apertures actually has a lower latency than the 80×80 sub-apertures case. We believe this is due to the reduced number of pixels per sub-aperture when using 90×90 sub-apertures. Because all of the SH-WFS tests use the same simulated camera image dimensions of 800×800 pixels, the 90×90 sub-apertures case is only using 8×8 pixels per sub-aperture compared to 10×10 pixels per sub-aperture for the 80×80 case. This reduces the perceived latency in two ways. First, by the reduced pixel processing required and secondly by the fact that the latency is measured as the time from when the last pixel arrives and due to the way this is measured, the timestamp is taken when the full 800×800 image has arrived which is later than when the processing completes 90×90 sub-apertures case.

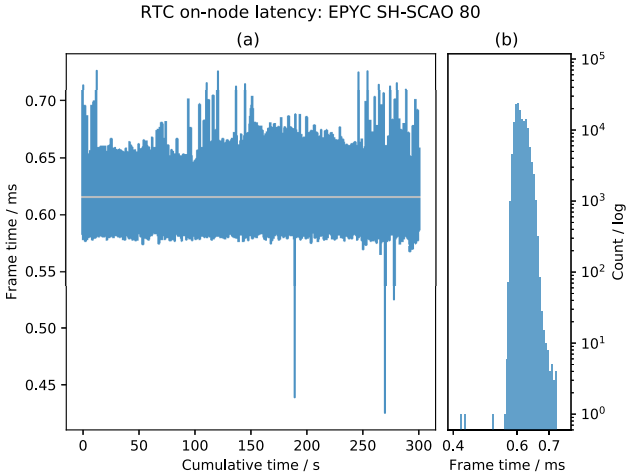

Figure 9. Frame time results for an SCAO set-up using a Shack–Hartman type WFS slope calculation with 80 sub-apertures across the pupil for an AMD EPYC system as described in Section 2.4. Results shown are for 1.5×10^5 iterations at 500 Hz for a total time of 300 s. The mean latency is 616 ± 17 µs.

If the latencies for the Pyr-WFS in Table 2 are compared directly with the latencies from the SH-WFS for the same WFS dimensions, we see that the SH-WFS overall results in reduced latencies. This is a result of the reduced pipelining efficiency of the Pyr-WFS versus the SH-WFS as described in Section 2 and shown in Fig. 4.

2.4 Other many-core CPU systems

Most of the results presented in this report are obtained from Intel Xeon Phi CPU systems as described in Jenkins et al. (2018). However, the Xeon Phi platform has been discontinued and so it is unlikely to be considered as a candidate for real AO RTC hardware. One of the main reasons for choosing a CPU-based RTC is that the software and the optimizations made for many-core operation are not specific to a single CPU architecture or vendor. Fig. 9 shows frame time and latency results for DARC running an SCAO RTC on an AMD EPYC system with an attached simulated camera using the UDP pixel streaming method as described in Section 2. This can be compared directly to the Xeon Phi results shown in Fig. 5. There is 300 s worth of data corresponding to 1.5×10^5 frames at a frame rate of 500 Hz, the average latency is measured at 616 ± 17 µs.

The source code and RTC parameters are identical for the two different CPUs with the only differences coming from the compiler options and the configuration of the threading and NUMA aware memory allocation for the two different platforms. The EPYC system used for these results is an AMD EPYC 7351 dual socket system with 16 cores and 64GB of DDR4 2667 MHz memory per socket. The core topology of the EPYC CPUs is such that each CPU has four NUMA regions with each region having four CPU cores and 16 GB of memory each. The RTC software uses the NUMA information of the CPU to allocate memory for the RTC control matrix on the nodes relevant to each CPU core. The Kernel and OS is tuned in a similar way to the Xeon Phi with the major differences being the OS, Ubuntu 16.04 for EPYC versus CentOS for the Xeon Phi, and simultaneous multithreading (Hyper-Threading) is turned on for the EPYC system as it provides better performance and allows eight threads per NUMA node.

The maximum memory bandwidth of the EPYC system is 350 GBs^{-1} by having eight channels of DDR4 memory per socket and utilising NUMA aware software. This is less than the measured memory bandwidth of 480 GBs^{-1} of the Xeon Phi 7250. It is therefore expected that the performance of the EPYC will be less than that of the Xeon Phi for the memory bandwidth bound RTC operations. However, these results show that the software can be readily used on different CPU platforms and that performance is as expected based on the knowledge that the main RTC operations are memory bandwidth bound.

Other multisocket CPU systems would also be suitable for ELT-scale AO RTC such as the Intel Xeon Scalable processors which in a quad socket configuration can provide higher maximum memory bandwidth than the Xeon Phi when the NUMA regions are taken into account. The multisocket systems also benefit from generally running at a higher base CPU frequency than the Xeon Phi and so their single threaded performance is better. A dual-socket EPYC system with the required memory bandwidth can be purchased for a similar price to the Xeon Phi, making it the most likely substitute. For the Intel Xeon Scalable processors, due to their reduced memory channels per socket, a quad socket system would be required to match the memory bandwidth and this can increase the per node costs to over 4 × that of comparable EPYC or Xeon Phi systems.

Next generation AMD EPYC processors will introduce a new architecture (Papermaster 2018) that simplifies the core topology of the system and will be built on a smaller 7 nm process node to provide better energy efficiency. This involves introducing a 9-die architecture which includes eight compute chiplets and a single I/O interface die such that each CPU core can access all memory channels equally. This is different to the current design where each of the four NUMA regions has eight cores and two memory channels each and so only those eight cores can access the full bandwidth of those two channels. This should reduce the relative complexity of NUMA memory management and allow more efficient interleaving of memory over all eight memory channels which will likely reduce the latency of the EPYC results shown in Fig. 9. The memory for the next-generation processors is also likely to be clocked faster, at up to 3200 MHz compared to the current maximum of 2667MHz, increasing memory bandwidth.

3 PROTOTYPING AN MCAO AND LTAO RTC

As mentioned in Section 1.1, MCAO and LTAO generally differ from SCAO by the number of WFSs and DMs used. Therefore, in the context of the RTC they are both much more computationally demanding than the simple SCAO case. When prototyping an RTC architecture for MCAO and LTAO, we decided to design it based on the ELT first-light instruments, the Multiconjugate Adaptive Optics RelaY (MAORY; Ciliegi et al. 2018) and the HARMONI LTAO mode (Neichel et al. 2016). For both of these instruments, the most demanding aspects will be the reconstruction of six laser guide star (LGS) WFSs operating at 500 Hz, the parameters for these instruments and a comparison with an SCAO system are shown in Table 1. Targeting proposed ELT MCAO and LTAO instruments allows us to demonstrate more realistic test cases and puts better constraints on the design of the architecture.

One of the main benefits of designing a CPU-based RTC lies in the flexibility and generality that the CPU architecture provides. The product of this is that the software optimizations and modifications detailed in Jenkins et al. (2018) for SCAO can be readily applied to different AO types such as MCAO and LTAO. Therefore, the architecture we propose for these AO regimes is an extension of the SCAO case by scaling the software and hardware to match the increased computational complexity.

Table 1. A comparison of the specifications of ELT-scale SCAO, MCAO, and LTAO, the values are the most current known specifications for the HARMONI SCAO mode, the MAORY MCAO mode and the HARMONI LTAO mode respectively.

Mode	SCAO	MCAO	LTAO
Target frame rate (Hz)	1000	500	500
LGS number	0	6	6
LGS sub-aperture geometry	N/A	80 × 80	80 × 80
LGS pixel geometry	N/A	10 × 10	10 × 10
LGS total sub-apertures	N/A	4616 × 6	4616 × 6
LGS image format	N/A	800 × 800	800 × 800
NGS number	1	3	1
NGS type	Pyramid	SH-WFS	SH-WFS
NGS sub-aperture geometry	100 × 100	2 × 2	2 × 2
NGS pixel geometry	N/A	100 × 100	100 × 100
NGS total sub-apertures	4616	4 × 3	4
NGS image format	240 × 240	240 × 240	240 × 240
DM number	2	3	1
Total DM modes	5316 + 2 +176	5316 + 2 +2 × 500	5316 + 2 +6 × 2

For the MAORY MCAO and HARMONI LTAO parameters detailed in Table 1 the computational demands for each LGS WFS are similar to those of the SH-WFS SCAO case tested in Section 2.3 with slightly more DM actuator DoF but targeting a reduced frame rate. We therefore decided that for each LGS WFS the processing of the WFS images to reconstructed wavefronts can be done in a very similar way to the SCAO procedure in Section 2.3 by processing a single LGS WFS per processing node.

The NGS parameters demand far fewer computational resources than the SCAO case demonstrated on a single node in Section 2.3, therefore we propose that the NGS WFS processing, three NGS for MAORY and one for HARMONI, can be achieved by a single instance of DARC on a single processing node. The partial DM commands resulting from these calculations then need to be summed together to produce a single DM command vector for all of the required DMs. This will be achieved by having a separate 'master' processing node which can receive partial DM commands from the reconstruction nodes and perform any post-processing that may be required before delivering the final actuator commands to the DMs.

Fig. 10 shows the prototype architecture for MCAO/LTAO RTC using the ideas described above. There are a total of seven reconstruction nodes to process all the WFSs required for the either the MAORY MCAO case or the HARMONI LTAO case. An 8th master node is used for summing the partial results from each reconstruction node and processing them for sending to the DMs. Due to the way the ELT M4 will operate (Xompero et al. 2018), the correction applied by M4 will not necessarily be the same as that which is delivered to M4 from the RTC. It will therefore be necessary for the RTC of an ELT instrument to receive feedback from M4 such that it can incorporate the actual correction applied for the previous iteration. In our prototype architecture, this will be processed by the master processing unit, shown in Fig. 10.

Fig. 11 shows a lab test set-up derived from the prototype architecture shown in Fig. 10. It shows the simulated camera delivering the required camera streams to each reconstruction node over a high-speed network. The master processing node communicates with the reconstruction nodes over a separate high-speed network on the same physical switch but using a different VLAN and different

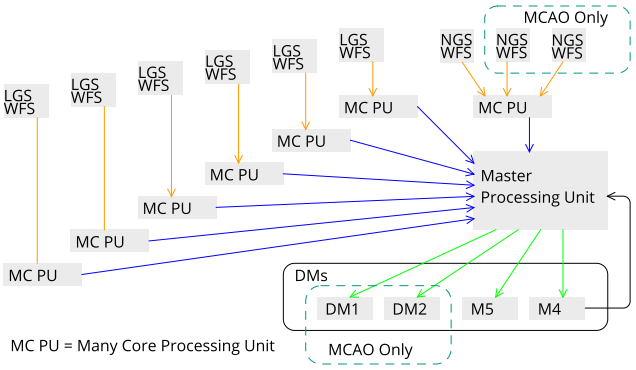

Figure 10. The proposed architecture for the MCAO and LTAO multi node RTC. The six LGS WFSs are each processed by a single many-core node. The NGS are all processed on the same node, three for MAORY and one for HARMONI. The master node sends out the final DM commands once it has finished summing and processing the partial vectors. The master node should also able to receive feedback from the ELT M4 to integrate the actual M4 shape used in the next command.

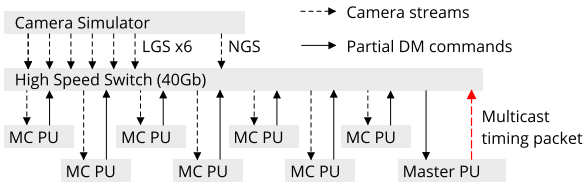

Figure 11. The lab test set-up for the MCAO and LTAO architecture. The simulated camera streams six LGS WFSs and either one or three NGS WFSs through a high speed switch to the processing units. The many-core processing units send their partial DM commands calculated from the individual WFSs to the master node for summing and further processing. For the tests to determine the overall latency of the system, the master node multicasts a timing packet for all other nodes to time stamp the end of frame.

network interconnects to the camera streams. For our lab test set-up, the full RTC latency is calculated on each reconstruction node by measuring the time between when it receives the last pixel from its camera stream to the time it receives a timing packet from the master which is multicast to the reconstruction nodes when it has completed the current frame. As the cameras use the proposed ESO MUDPI packet they stamp each image with a frame number which is then propagated through to the master and back through the timing packet to be able to match the correct DM command to the camera frames.

3.1 Results of testing the prototype

When testing the MCAO prototype described in Section 3 certain considerations needed to be made with regards to the timing of the individual frames. Ideally, we would want to measure the time from last pixel into the RTC until the time the DM command is ready, however we discovered that for a multinode CPU-based architecture it is difficult to synchronise the clocks between nodes with enough precision using our available hardware such that timestamps generated on each node can be directly compared. Instead, we have to rely on only comparing timestamps generated on individual nodes and so the full RTC latency is calculated as the time between a reconstruction node receiving the last pixel from its camera stream and the time it receives a timing packet from the

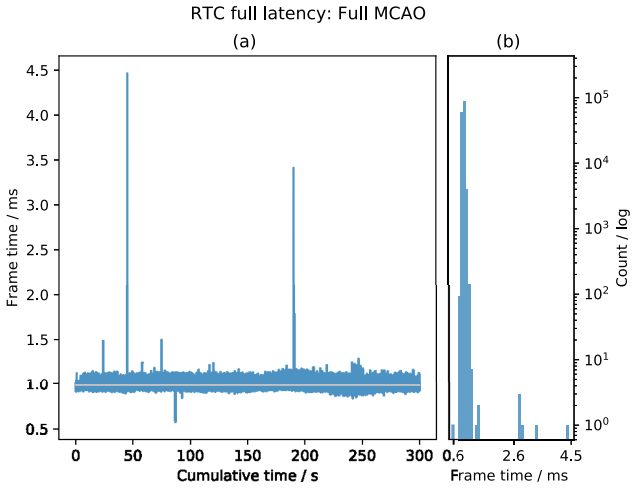

Figure 12. Latency results for the full MCAO set-up as described in Section 1.1 and Table 1. This is for 1.5×10^5 iterations at 500 Hz corresponding to a total cumulative time of 300 s. This is a measure of the time between when the last pixel arrives at a reconstruction node and when it receives the timing packet from the master node. The large outliers are result of delays from the CPU-based simulated cameras, which is compounded by the fact that this timing data includes delays from all seven simulated camera streams.

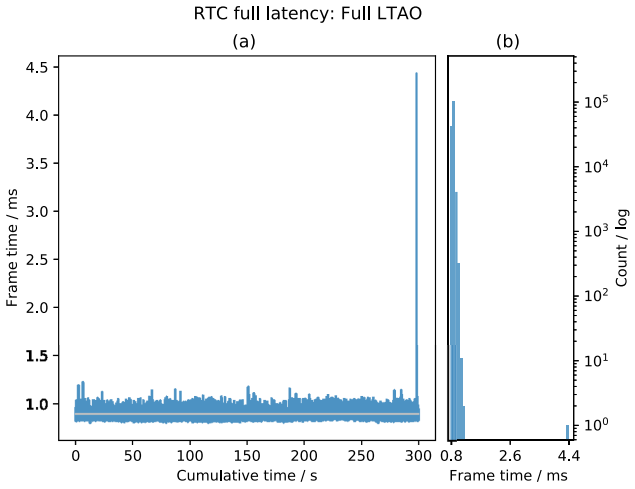

Figure 13. Latency results for the full LTAO set-up as described in Section 1.1 and Table 1. This is for 1.5×10^5 iterations at 500 Hz corresponding to a total cumulative time of 300 s. This is a measure of the time between when the last pixel arrives at a reconstruction node and when it receives the timing packet from the master node. The large outliers are result of delays from the CPU-based simulated cameras, which is compounded by the fact that this timing data includes delays from all seven simulated camera streams.

master node indicating that the final DM command is ready, which does result in a slightly pessimistic measurement.

Ideally, the full RTC latency would be measured externally with a device that is able to record the time between when the camera finishes sending a frame and the time when the corresponding DM command is received from the master node. This process would make the RTC latency measurement easier however the method described above is no less valid as a measure of the RTC latency.

For all the MCAO and LTAO tests described in this report, the system architecture used is shown in Fig. 10 and network interconnects are as shown in Fig. 11. Each of the seven reconstruction nodes uses a single instance of the DARC software to receive an 800×800 pixel camera stream which it process from pixels to a partial DM command for that WFS. This processing is done in a very similar way to the SH-WFS SCAO case as described in Jenkins et al. (2018) and in Section 2.3, the main differences are the way in which the reconstruction matrix is constructed and that the DM software library is used to send the partial DM to the master node. The master node itself runs a separate instance of DARC which receives the partial DM commands, sums them together, processes the result and once finished it sends a timing packet to the other nodes.

Figs 12 and 13 shows RTC latency plots for the MCAO and LTAO test cases, respectively. They show the timing data from one of the reconstruction nodes of the seven during full operation, the timing data also includes the transmission and receiving time of the DM timing packet and can therefore be considered slightly pessimistic. The specification used for each are shown in Table 1 and the mean latency of the MCAO case is 985 ± 33 µs and 894 ± 29 µs for the LTAO case. It can be seen that there are two major outliers in the latency for the MCAO test and one for the LTAO test which are a result of delays introduced from the CPU-based simulated cameras. The number of frames losses however is acceptable considering that there is at worst one frame drop per 150 s total integration time.

As well as testing the full eight node MCAO RTC we also tested the architecture with different numbers of reconstruction

Table 2. Latency, RMS jitter and largest outliers results for all of the data presented in this report. For all results other than LTAO with buffer swap the (a) columns correspond to results from 1.5×10^5 continuous iterations at 500 Hz for a total time of 300 s for each test case. The (b) columns correspond to results from a subset of no less than 2×10^4 continuous iterations chosen from the larger (a) data sets for a total time of 40 s each. The (b) column subsets were chosen to avoid any large outliers that result from simulated camera delays to give a better representation of the 'steady' latency. The LTAO with buffer swap results are for 1.5×10^4 continuous iterations at 500 Hz for a total time of 30 s, there are no 'steady' results for this case as the outliers here are a result of the parameter swap itself and not from an external factor.

AO mode	Mean latency (µs)	RMS jitter (µs)		Largest outlier (µs)	
		(a)	(b)	(a)	(b)
Pyr-WFS 80 × 80	609	11	11	898	653
Pyr-WFS 90 × 90	796	16	16	1051	862
Pyr-WFS 100 × 100	998	10	9	1667	1049
Pyr-WFS 110 × 110	1209	12	12	1433	1292
Pyr-WFS 120 × 120	1450	8	8	1886	1508
SH-WFS 74 × 74	357	18	16	495	436
SH-WFS 80 × 80	511	15	15	754	589
SH-WFS 90 × 90	497	14	14	1237	559
SH-WFS 100 × 100	778	12	11	1314	821
SH-WFS 110 × 110	1001	14	14	1397	1065
Full MCAO	985	33	29	4465	1235
MCAO telemetry	1085	32	30	2988	1466
MCAO POLC	1090	45	44	2880	1312
MCAO 6 LGS	992	47	46	3498	1143
MCAO 5 LGS	979	46	44	2870	1261
MCAO 4 LGS	969	45	45	3305	1285
MCAO 3 LGS	943	43	42	2817	1182
MCAO 2 LGS	951	42	43	2858	1119
Full LTAO	894	29	28	4434	1174
LTAO with buffer swap	1034	31	–	1949	–

nodes combined with the master node. Table 2 shows the results for the MCAO case where there are less than the full number of reconstruction nodes for the MAORY specification. Results are shown for the cases where there are 2–6 LGS reconstruction nodes feeding partial DM commands to the master node. As can be seen from the results, the total latency varies only slightly by the reduction in processing nodes, showing that the solution is scalable at least up until the desired number of nodes for ELT-scale MCAO.

3.2 Effect of on-the-fly changes to RTC parameters on latency

An important aspect of any real on-sky RTC is the ability to change parameters during operation, for example to update the matrix used in the reconstruction process or to update the reference centroids. DARC has the ability to set parameters through two different means, either through a command line utility called 'darcmagic' which can be used to change simple parameters such as string and scalar values, or through a PYTHON interface which can also change more complex parameters such as the arrays and matrices. DARC uses a double buffer approach to handle parameter switching, the buffers contain all the necessary parameters for RTC operation and whilst one of the buffers is read by the RTC during operation the second can be modified to include any required new values without affecting the running processes.

Once the necessary changes have been made to the second buffer, DARC performs a buffer swap which causes it to start reading values from the second buffer instead of the first. The process that DARC uses to handle a buffer swap involves a flag in the main processing loop which instructs the first thread that begins a new frame that a buffer swap is required and that thread performs some checks, updates some information, and then replaces the buffer pointer. Because all of the processing threads will read from the buffer this needs to be thread safe and so all other threads are temporarily blocked while the buffer is swapped.

The double buffering of the parameters reduces the effects of a parameter change by allowing the majority of the change to happen during general operation without affecting latency. However, on a platform like the Xeon Phi which excels on multithreaded performance, the single threaded buffer swap can have a noticeable impact on the latency of the frame during the swap. Fig. 14 shows the effect of a buffer swap on an SCAO system set-up as described in Table 1. The first two latency spikes seen are due to changing the control matrix of size 5318 × 9232 which takes approximately 10 s for the transfer to happen. The other latency spikes are due to a periodic change of a single value parameter every five seconds. The size of the latency spikes shows that for different size parameters the amount of jitter introduced is the same and the spike only occurs once the internal RTC buffer is actually swapped.

Fig. 15 shows the latency for an LTAO type system set-up as described in Table 1 whilst a buffer swap is set to occur on one of the reconstruction nodes every 5 s. There is a clear impact on the latency which corresponds to a ≈800 µs spike to latency when the buffer swap occurs. Here, the relative size of the latency spikes is increased from the SCAO case above as this is the full LTAO RTC system as described in Section 1.1. This essentially results in a frame drop for every buffer swap due to the average latency being 1034 µs for this particular case.

We believe that the relatively large effect of a buffer swap on the latency of the LTAO system in this case is partly caused by the fact that the Xeon Phi has relatively poor single threaded performance and partly because of the need for all reconstruction nodes to be synchronized by the master node, compounding any adverse

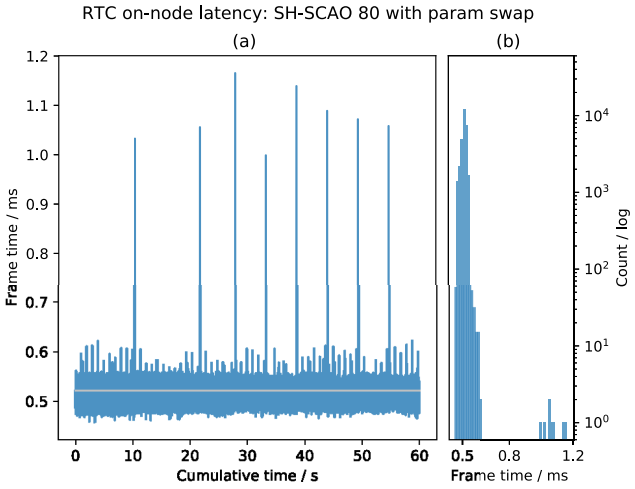

Figure 14. Latency results for an SCAO set-up as described in Section 2.3 whilst two types of buffer swap are performed as described in Section 3.2. As can be seen, the buffer swap causes a ≈650 µs spike to the latency whenever it is performed. This is for 3×10^4 iterations at 500 Hz corresponding to a total cumulative time of 60 s to highlight the 10 s transfer time of the first two parameter changes of the control matrix.

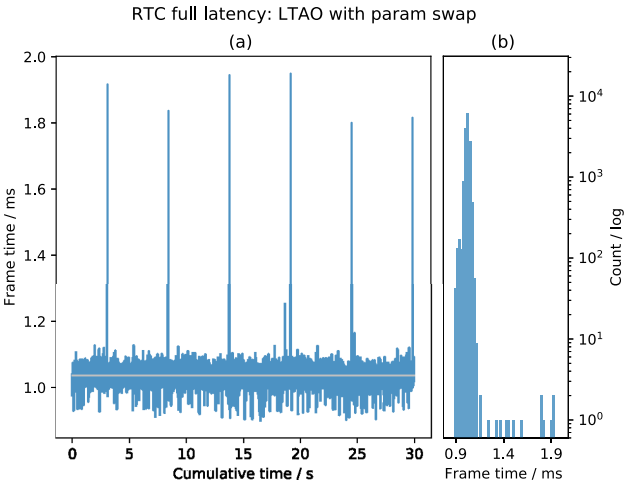

Figure 15. Latency results for an LTAO set-up as described in Section 1.1 whilst a periodic buffer swap is performed as described in Section 3.2. As can be seen, the buffer swap causes a ≈800 µs spike to the latency whenever it is performed. This is for 1.5×10^4 iterations at 500 Hz corresponding to a total cumulative time of 30 s to highlight the disturbance every 5 s.

effects of the swap. The parameter change performed here was for a single valued scalar parameter, however due to the double buffered approach of DARC, changing more complex parameters such as arrays or matrices should not have any more impact on the latency as all copying of data can occur concurrently with RTC operations.

3.3 Effect of streaming RTC telemetry on latency

Another very important aspect of AO RTC operation involves the streaming of telemetry during operation, either for concurrent processing so as to update the reconstruction matrices or reference centroids or purely for saving data to disc for later analysis. DARC employs circular buffers to store telemetry when requested during operation and also has the capability to read from these buffers and send the telemetry to wherever is necessary. All the timing data

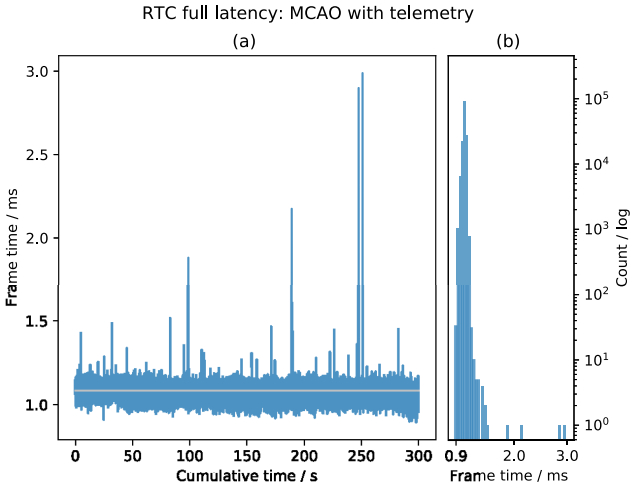

Figure 16. Latency results for an MCAO set-up as described in Section 1.1 whilst both centroid and DM command telemetry is taken for all nodes as described in Section 3.3. This is for 1.5×10^5 iterations at 500 Hz corresponding to a total cumulative time of 300 s.

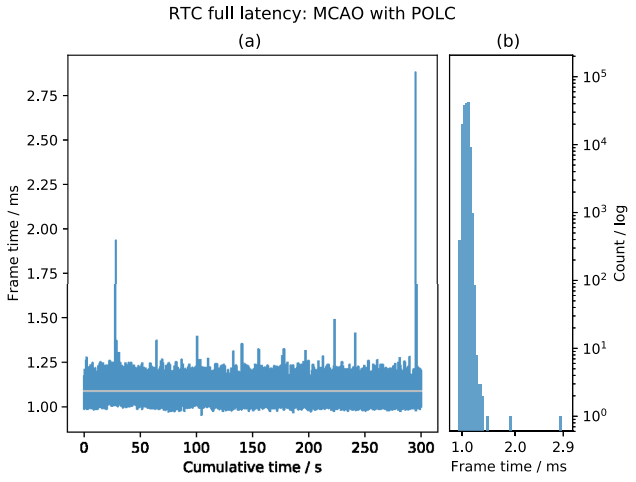

Figure 17. Latency results for an MCAO set-up as described in Section 1.1 whilst implicit POLC is computed on the master processing node as described in Section 3.4. This is for 1.5×10^5 iterations at 500 Hz corresponding to a total cumulative time of 300 s.

used in this report is gathered by using the telemetry streaming functionality of DARC. There are three buffers which are read for every timing measurement used in this report; an RTC time buffer which stores frame times, an RTC status buffer which stores various status information and and RTC DM time buffer which stores the timing data for the receipt of the timing packet from the master node.

The status buffer is used to retrieve the timestamps for when the last pixel has arrived and also the timestamp for when each node has delivered its partial DM command. The DM time buffer is populated by a process which listens for packets from the master node and takes a time stamp on arrival. A common iteration number between the two buffers originating from the simulated camera is used to synchronize the data. In this way, we can match up the DM command timing packet sent from the master to the image frame received from the simulated camera and calculate the full RTC latency.

Fig. 16 shows the RTC latency for an MCAO set-up for a case when slope telemetry and partial DM command telemetry is also streamed from the reconstruction nodes during operation. The mean latency is measured at 1085 ± 32 µs and there are several outliers which result from delays due to the simulated camera streams. There are also a number of relatively small outliers in this data, <1.5 ms, compared to the case without slope and DM telemetry which results from the taking of telemetry itself.

3.4 Effect of pseudo-open loop control on latency

All of the types of AO used in this report would generally be used in closed-loop operation, that is, the atmospheric wavefronts are corrected before the residual phase error is measured by the WFSs. This approach means that the wavefront phase errors measured by the WFSs are smaller than those measured in open loop and so the WFSs can be tuned for finer precision. Also, during closed-loop operation, errors in the measurement, correction, and reconstruction can be dynamically removed by the feedback of the system. The downside to closed-loop AO is that the slopes measured by the WFS no longer give a measurement of the actual atmospheric wavefront phase. The slopes measured in open loop can be used to retrieve information about the atmospheric conditions which are required for reconstruction algorithms that take the current atmospheric statistics into account.

To get around the lack of open-loop slopes in closed-loop operation, it is possible to reconstruct pseudo-open loop (POL) (Piatrou & Gilles 2005) slopes from the DM commands and resulting closed-loop slopes. For reconstruction algorithms that rely on POL control (POLC), there are two ways in which the POL slopes can be incorporated into the final reconstruction result. They can either be calculated explicitly and used directly in the algorithms or the effects of POLC can be incorporated into the final reconstruction implicitly without first calculating the actual POL slopes. The benefits of implicit POLC are massively reduced computational requirements. Explicit POLC requires the POL slopes to be computed before the MVM to calculate DM commands can be computed, however implicit POLC only needs the final DM command as calculated by the master node and so the POLC computation can be done there.

We have implemented the implicit POLC calculation for the MCAO and LTAO operation of DARC on the master node. The POLC is calculated for the next frame after the DM command is ready and so it should have minimal impact on the overall latency. The summing of partial DM commands on the master node is calculated by a single thread and so there are enough computational resources remaining to calculate the implicit POL, which is a single MVM, without affecting latency. Fig. 17 shows the RTC latency for an MCAO set-up for a case when the master node is performing POLC computation using 32 threads. The mean latency is measured at 1090 ± 45 µs and there is a single large outlier which results from delays due to the simulated camera streams.

4 CONCLUSIONS

We have presented an update on the work and results in Jenkins et al. (2018) and demonstrated a prototype many-core CPU RTC architecture for MCAO and LTAO which builds on the previously presented work. We have shown that latencies of <1000 µs can be achieved for both MCAO and LTAO using this architecture with CPU-based simulated cameras. We have demonstrated that running the DARC RTC software on Xeon Phi processors can achieve ELT SCAO latencies of less <600 µs for Shack–Hartman WFS

processing and <800 µs for Pyramid WFS processing. We have also demonstrated latencies of less <700 µs for Shack–Hartman WFS processing on an AMD EPYC system, showing the generality of the software and techniques. The effects of parameter switching, telemetry streaming and implicit POLC computation have also been shown for the MCAO and LTAO cases with minimal effects on the overall latency. The CPU-based simulated cameras have been shown to introduce large outliers in the latency distributions which are an unavoidable consequence of the hardware used.

As shown in Jenkins et al. (2018), the Xeon Phi processor is an example of a many-core CPU system with high-bandwidth memory and so the software and system architectures described in this report are readily transferable to other many-core CPU systems with the required computational and memory bandwidth specifications. More general many-core CPU systems could also achieve better latency and jitter due to the relatively weak single threaded performance of the Xeon Phi which greatly effects serial processes such as receiving pipelined pixel streams over network interfaces.

ACKNOWLEDGEMENTS

Real-time adaptive optics work by the Durham group is supported by the European Union Horizon 2020 funded GreenFlash project, Identification 671662, under FETHPC-1-2014, the UK Science and Technology Facilities Council consolidated grant ST/P000541/1, and an Science and Technology Facilities Council PhD studentship, award reference 1628730.

REFERENCES

AravisProject 2018, Aravis. Available at: https://github.com/AravisProject/aravis

Assémat F., Gendron E., Hammer F., 2007, MNRAS, 376, 287

Babcock H. W., 1953, PASP, 65, 229

Barr D., Basden A., Dipper N., Schwartz N., 2015, MNRAS, 453, 3222

Beckers J. M., 1988, in Ulrich M.-H., ed., European Southern Observatory Conf. and Workshop Proc. 30. European Southern Observatory, Garching bei Munchen, Germany, p. 693

Biasi R. et al., 2016, in Marchetti E., Close L. M., Véran J.-P., eds, Proc. SPIE Conf. Ser. Vol. 9909, Adaptive Optics System V. SPIE, Bellingham, p. 16

Ciliegi P. et al., 2018, in Close L. M., Schreiber L., Schmidt D., eds, Proc. SPIE Conf. Ser. Vol. 10703, Adaptive Optics Systems VI. SPIE, Bellingham, p. 1070311

Correia C., 2018, Architecture of ELT 1st light instruments' Hard Real Time Computing Facility with Xeon-Phis. Chimie Paris Tech, Paris

Downing M. et al., 2018, in Close L. M., Schreiber L., Schmidt D., eds, Proc. SPIE Conf. Ser. Vol. 10703, Adaptive Optics Systems VI. SPIE, Bellingham, p. 107031W

Foy R., Labeyrie A., 1985, A&A, 152, L29

Fried D. L., 1966, J. Opt. Soc. Am., 56, 1372

Fried D. L., 1982, J. Opt. Soc. Am., 72, 52

Fugate R. Q. et al., 1991, Nature, 353, 144

Jenkins D. R., Basden A., Myers R. M., 2018, MNRAS, 478, 3149

Johns M., Angel R., Shectman S., Bernstein R., Fabricant D., McCarthy P., Phillips M., 2004, Status of the Giant Magellan Telescope (GMT) project. Univ. Arizona Press, Tucson, p. 5489

Kerrisk M., 2018, PACKET(7) Linux Programmer's Manual. Available at: http://man7.org/linux/man-pages/man7/packet.7.html

Neichel B. et al., 2016, in Marchetti E., Close L. M., Véran J.-P., eds, Proc. SPIE Conf. Ser. Vol. 9909, Adaptive Optics System V. SPIE, Bellingham, p. 15

Papermaster M., 2018, AMD Next Horizon. Available at: https://www.amd.com/system/files/documents/next_horizon_mark_papermaster_presentation.pdf

Piatrou P., Gilles L., 2005, Appl. Opt., 44, 1003

Ragazzoni R., 1996, J. Mod. Opt., 43, 289

Schreiber L. et al., 2018, in Close L. M., Schreiber L., Schmidt D., eds, Proc. SPIE Conf. Ser. Vol. 10703, Adaptive Optics Systems VI. SPIE, Bellingham, p. 107031Y

Schwartz N. et al., 2018, in Close L. M., Schreiber L., Schmidt D., eds, Proc. SPIE Conf. Ser. Vol. 10703, Adaptive Optics Systems VI. SPIE, Bellingham, p. 1070322

Spyromilio J., Comerón F., D'Odorico S., Kissler-Patig M., Gilmozzi R., 2008, The Messenger, 133, 2

Stepp L. M., Strom S. E., 2004, in Ardeberg A. L., Andersen T., eds, Proc. SPIE Conf. Ser. Vol. 5382, Second Backaskog Workshop on Extremely Large Telescopes. SPIE, Bellingham, p. 67

Thatte N. A. et al., 2014, in Ramsay S. K., McLean I. S., Takami H., eds, Proc. SPIE Conf. Ser. Vol. 9147, Ground-based and Airborne Instrumentation for Astronomy V. SPIE, Bellingham, p. 11

Xompero M., Briguglio R., Pariani G., Riccardi A., 2018, in Close L. M., Schreiber L., Schmidt D., eds, Proc. SPIE Conf. Ser. Vol. 10703, Adaptive Optics Systems VI. SPIE, Bellingham, p. 1070345

This paper has been typeset from a TEX/LATEX file prepared by the author.